DOI: 10.2298/CSIS110310062L

A Case Study on REST-Style Architecture for Cyber-Physical Systems: Restful Smart Gateway

Qiang Li [1, 2], Weijun Qin[1], Bing Han[3], Ruicong Wang[3], and Limin Sun[1]

[1] Institute of Software, the Chinese Academy of Sciences,
Beijing 10090, China
{ liqiang, qinweijun, sunlimin }@is.iscas.ac.cn
[2]Graduate University of Chinese Academy of Sciences,
Beijing 10090, China
{ liqiang }@is.iscas.ac.cn
[3]School of Software and Microelectronics, Beijing,
102600 Beijing, China
{hanbing, wangruicong}@is.iscas.ac.cn

Abstract. Due to several key factors, Cyber-physical systems (CPS) pose great challenges in software system design, which are dynamic composition, heterogeneous, adaptation and uncertain in environmental factors. In this paper we present our research on the development of REST-style architecture for CPS. We propose a path towards solving requirements of CPS architecture through Restful principles. By using this architectural style, we have built a prototyping system called the restful smart gateway, which seamlessly integrates conceptual and physical resources into the Web and scale better. Some experiments on the smart gateway are given to illustrate its performance.

Keywords: cyber-physical systems, REST, architecture, smart gateway, wireless sensor network.

1. Introduction

Cyber-physical systems (CPS) [2] are physical and engineered systems whose operations are monitored, coordinated, controlled and integrated by a computing and communication core. CPS bridges the virtual world of computing and communications and the real world, which lead to enormous societal impact and economic benefit. CPS pose great challenges [1] in software system design, due to several key factors: (1) physical devices expose their functionalities to the outside with networking capability, leading to the complexity of the system increases exponentially (2) the cyber and physical worlds are tightly integrated, which requires systems should tolerate the failures of components and uncertainty or the noise in the physical environment, and (3) components are inherently heterogeneous. Given

above-mentioned challenges designing CPS, one system design solution will not meet the requirements. Rather, what we need is a basic system software architecture [3] upon which services can easily be designed, deployed and composed on demand by individual applications, in a manner that satisfies specific safety, security, reliability, efficiency and predictability requirements, while still remaining within the bounds of given hardware capabilities.

In this paper we propose to use REST-Style architecture [4] as design principles, to guide developing of CPS. The Representational State Transfer (REST) style provides a set of architectural constraints that emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems. As demonstrated by the success of the Web (using REST-style architecture guide), loosely coupled approaches possess high scalability and robustness - which are fundamental properties for building a worldwide network of devices. Furthermore, the real value of such applications comes from the sharing and integration of data among heterogeneous devices. Based on these considerations, our proposal is about CPS designing and developing under the guiding of REST-Style architecture, which seems to be the primary choice of system design is to ensure reliable, safe, efficient and predictable behavior of the applications.

The contribution of this paper is given as follows. On one hand it proposes to use REST-Style architecture as the basis for systems developing that meet the requirements for CPS architecture. On the other hand it proposes a case study of the smart gateway, which is lightweight, scalability and extensible software components that enable Web-based interactions with all kinds of embedded devices. For CPS, the role of smart gateways simplifies greatly the process to export data and functionality of the physical devices to be provided to end users on the web. Using our prototype system to illustrate that REST-Style architecture is suitable for CPS.

In this paper, we present REST-Style architecture as an approach to design and developing CPS. In section 2, we present a survey of related works. In section 3, we extracted from the existing work the requirements for CPS and described the main characteristic of REST-Style architecture. In section 4, we present the implementation of prototype system, while Section 5 deals with a performance evaluation of the whole infrastructure. Finally, Section 6 discusses future work and concludes the paper.

2. Related Work

Lee [1] examines the challenges in designing CPS by considering whether current computing and networking technologies can support CPS design. The CPS Steering Group [2] present a complete description of issues related to CPS. In the work [3], Raj et al. further elaborate a multitude of technical challenges about the design, construction and verification of CPS, which

must be addressed by a cross-disciplinary community of researchers and educators. Most of the works on CPS raise design challenges or design issues such as dynamic composition, and high confidence software design [12-15]. Ying Tan [6, 7] has employed CPS architecture. These works define control components which determine and provides necessary functionality, and propose spatio-temporal event model. Abdelwahed and his colleagues [8] present an approach to designing high confidence software for CPS. They present four design methods; model-driven system execution modeling, model-based diagnosis, online control, fault-adaptive control, and an architecture for CPS. These are the motivations for our research, which try to gain a general architecture for CPS.

Differs from all of these architectures about CPS, our work profoundly is influenced by the concept of the REST principles, which guide designing and developing of CPS. The notion of REST has been conceptualized in Roy Fielding's PhD thesis [4]. Erik Wilde [5] proposes to put things to REST, which combine today's Web content and the integration of physical things. This is one of the first projects that envision the notion of the Web of Things. Following this direction, Stirbu [20] also enables heterogeneous sensor devices in the Web, but it focuses mainly on the discovery of these devices. TinyREST [19] offers a Restful gateway that has some similarities to our implementation but it violates REST principles by introducing the extra verb SUBSCRIBES. In addition, no evaluation of the system is provided. The work [9] is also about the REST-Style architecture, but only to remote control of sensors over a Wireless Sensor Network and the Internet. The work [10, 11] is similar to ours; develop an open web-based architecture for controlling the behavior of systems composed of static sensors, mobile sensors and unmanned vehicles.

Unlike these previous works, we implement a prototype system about the restful smart gateway, which handles HTTP requests and thus abandons the idea of having an HTTP server directly on the device. The smart gateway is lightweight, scalability and extensible software components that enable Web-based interactions with all kinds of embedded devices.

3. REST-Style architecture for CPS

3.1. Requirements for CPS Architecture

Most of the works on CPS raise design challenges or design issues such as dynamic composition, and high confidence software design [12-15].In particular, we envision four distinct challenges in design systems that cover the most important use cases that should be supported by a suitable architecture:

- Composition. Cyber-physical systems are inherently heterogeneous not

only in terms of their components but also in terms of essential design requirements. This heterogeneity will lead to that systems don't behave well outside of a small operational envelope and that are hard to maintain.

- Systems integration. Since physical processes are coupling with computing and communication processes, system components designing becomes complex. Simply using current technology, it is difficult in integrating complex components into complex systems.
- Reliability. Since physical devices interface with the uncertainty and the noise in the physical environment, systems must tolerate or contain the failures of components in both the cyber and physical domains.
- Security and Privacy. Cyber-physical systems open up new threats: physical systems can now be attacked through cyberspace and cyberspace can be attacked through physical devices.

3.2. REST-Style Architecture

Based on existing solutions for CPS architectures (Section 2), along with the requirements for Cyber-physical systems (Section 3.1), we suggest using REST-style architecture, which seems to be the primary choice of system design to ensure reliable, safe, efficient and predictable behavior of the applications.

The Representational State Transfer (REST) [4] style is an abstraction of the architectural elements, providing a set of architectural constraints that emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems. The REST community has been working on refining the notions to create Resource Oriented Architectures (ROA), in order to provide and connect together services on the Web. Because of the underlying simplicity, scalability of this architecture, we believe that they could be adapted in order to interconnect the physical world. In particular, we envision three major points of REST:

- Data Elements. A resource is a key abstract concept in REST that is assigned a URI and then can be used on the systems. Any information that can be a resource: a document or image, even physical device. A resource representation is a sequence of bytes, plus representation metadata to describe those bytes. REST components perform actions on a resource by using a representation to capture the current or intended state of that resource and transferring that representation between components.
- Connectors. REST connectors provide a generic interface for accessing and manipulating the value set of a resource, regardless of how the membership function is defined or the type of software that is handling the request. REST components communicate by transferring a representation of a resource in a format matching one of an evolving set

of standard data types, selected dynamically based on the capabilities or desires of the recipient and the nature of the resource. Whether the representation is in the same format as the raw source, or is derived from the source, remains hidden behind the interface. The connectors present an abstract interface for component communication, enhancing simplicity by providing a clean separation of concerns and hiding the underlying implementation of resources and communication mechanisms.

- Components. The vision is to design a system as a collection of application-specific services and abstractions, which are automatically structured and deployed on a target platform according to constraints in terms of: (a) hardware capabilities, and (b) application requirements.

In conclusion, the central feature of REST architectural style is its emphasis on a uniform interface between components. By applying the engineering principle of generality to the component interface, the overall system architecture is simplified and the visibility of interactions is improved. Implementations are decoupled from the services they provide, which encourages independent evolution. However, the trade-off is that a general interface degrades efficiency, since information is transferred in a standardized form rather than one which is specific to an application's requirements.

4. A Case Study: Restful Smart Gateway

Here we describe how we implemented our smart gateway to illustrate that REST-style architecture is suitable for CPS. Due to the fact that most of the embedded devices that are used today to represent physical things do not offer TCP/IP networking by default but dedicated communication protocols and technologies, we develop a smart gateway, from the Web to the physical devices and vice-versa. We present in detail our implementation efforts in the following subsections by utilizing our framework.

4.1. Smart Gateway Architecture

The conceptual overview of the smart gateway architecture is shown in Fig. 1. Presentation module generates dynamically a representation of the available devices and their corresponding services to the Web, enabling the uniform interaction with them over a Restful interface. Device management is responsible for the management and control of embedded devices, including discovery devices, maintain device adding and delete devices. Data management incorporate data and command parse modules, which convert and parse data among different dedicated communication protocols.

Presentation module represents the access point to the smart gateway from the Web. A Web Server allows clients to interact with any available

devices and their corresponding services using any Web browser or the client. Physical devices and services can be accessible by clicking URL links. We implement the interaction with any resource through a Restful API, which is provided by a REST Engine [16]. Restlet [16] was used to implement the REST Engine, as it has very stable performance in Java environments. For resource presentations, we implement two kinds' types: XML and JSON.

Device management is responsible for the management and control of embedded devices by maintaining a device list. Device management maintains device by broadcasting periodically a message. If an embedded device appears in the scale of the smart gateway, it will attempt to connect to our smart gateway by sending a "HELLO" message. As soon as this message is received by the smart gateway, it is immediately "ACK" message. At that time, the smart gateway will add the newly-found device in the list of devices and sends a second "ACK" message to the device. If the smart gateway receives a response from the device, it will generate a single URL and the corresponding resource presentation. The device management broadcasting periodically a message, if the embedded device that depletion of energy or broken down cannot give a response to the smart gateway. If the smart gateway device management does not receive a response more than five minutes, device management will delete the item in list of devices and destroy the URL.

Data management incorporate data and command parse modules, which convert and parse data among different dedicated communication protocols. Presentation module will send this URL clicked by clients to the command parse module. The command parses module will analysis and parses the URL, and send a command to the sink. Sensors transmit data through IEEE802.15.4 to the sink. Sink receive data and transmit it to the data management by the serial port. The data parse module complete data parse and store it in the database or cache.

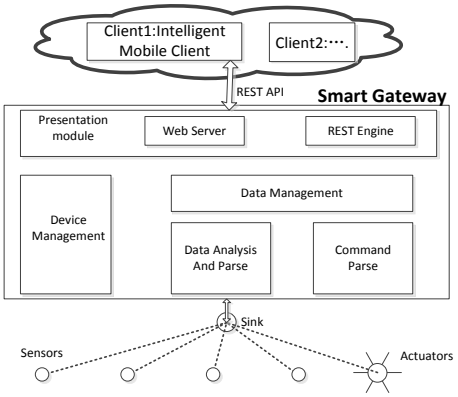

Fig. 1. Restful Smart Gateway Architecture

4.2. Restful Embedded Devices

We decided to select sensor motes to represent embedded devices in our implementation efforts, since they are very easy to program and they offer some basic sensing capabilities. Our version of restful smart gateway that handles HTTP requests and thus abandons the idea of having an HTTP server directly on the device. The reason is that improving the scalability of restful smart gateway with respect to concurrent requests on the same device, enhanced support for asynchronous communication for the values returned by the devices. To exchange sensor and actuator data between the device and the gateway, we using the device management and data management to guarantee the smart gateway return the currently stored representation of the device.

These sensor motes are equipped with a 250kbps, 2.4GHz, IEEE802.15.4-compliant Chip CC2420 Radio. IEEE802.15.4 is an IEEE standard that defines a MAC and PHY layer targeted to Wireless Sensor Networks (WSN). Fig. 2 shows the node of Telosb that we have used in the experiment.

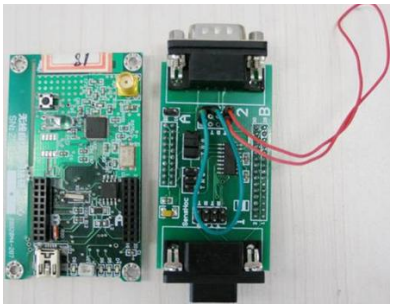

Fig. 2. The node of Telosb

We use two kinds of representations: XML and JSON.

- XML[18] is the default standard for structured information on the Web. Application scenarios can define their own schemas for XML, but in many cases standards or standards exist and can be reused. Depending on the type of resource, the representation must make the relevant aspects of the resource through XML
- For Web 2.0 applications, data is read from the server in JavaScript and then used to drive a dynamically updated Web page. The JavaScript Object Notation (JSON) [17] provides a better solution for JavaScript environments. This representation may be more limited than XML, but can be a good way to make resource data available for Web 2.0 applications.

In Table 1, we can see a general description of the Restful Web Services we created. SenCurrentRes predicates the root of current resources; SenActiveRes maintain the synchronization with the device list, predicating available resources; SenHistoryRes back given time history data;

SenCmdRes alter the state of physical devices. The first three resources have the same REST verbs, which GET is used to retrieve a representation of a resource. The fourth actuator has POST, which is a method to alter the state of a resource. All the resources have two kinds' types. Once the smart gateway receives a Restful HTTP request from a Web client, it makes the necessary validity checks and encapsulates the request as XML or JSON. It sends them to the sink, which is acting as a base station and directly connected to the smart gateway by serial port.

Table 1. A list of the RESTful devices

Resource URI	REST Verb	MIME Type	URL (the fixed prefix are http://localhost:port/wsn/)
SenCurrentRes	GET	XML/JSON	sensors/{sensorid}/{dataType}.{mediaType}
SenHistoryRes	GET	XML/JSON	sensors/history/{sensorid}/{dataType}/{fromTime}/{toTime}.{mediaType}
SenActiveRes	GET	XML/JSON	sensors/activeNode/all.{dataType}.{mediaType}
SenCmdRes	POST	XML/JSON	sensors/{sensorid}/command/{dataType}/interval

Fig. 3. The user interface of the client is showing the emperature of the surround environment which the device measured.

We have created the mobile phone application using the services provided by the restful gateway. The application combines physical and Web resources in the Restful API. The user interface of the application running the smart phone is shown as Fig. 3. The phone view is used to obtain current data from the wireless sensor network. The smart phone updates the current

environment information every ten seconds through sending GET URL requirement to the restful gateway. The gateway will give the response as the feedback to the phone. As shown in Fig. 3, we have chosen the temperature sensor whose device number is one to get current laboratory indoor environmental temperature. The real-time data view displays the current temperature information in the form of the compass.

5. Evaluation

Using the uniform interface would inevitably sacrifice efficiency specially compared with these gateway just transmit sensor data without using URL. Therefore, we have evaluated the performance of the prototype system through testing the amount of delay time from a request to the back response. As a compared target, we have also gain another delay time which through a normal gateway just translating sensor data to the Internet.

We have set the restful gateway and the service provider into the same personal computer. Around it, we deployed fixed scale of ten sensors. The sink node was plugged in one of the USB ports of the personal computer to serve as a base station, to forward IEEE802.15.4 wireless packets from/to the personal computer through the serial-over-USB port. Since the internet delay is dynamic, the client and the smart gateway using the same local IP address. We focus on gain the delay time from the smart gateway serial port reading date to the client getting the response. We performed ten times test and gain their average time in the experiment.

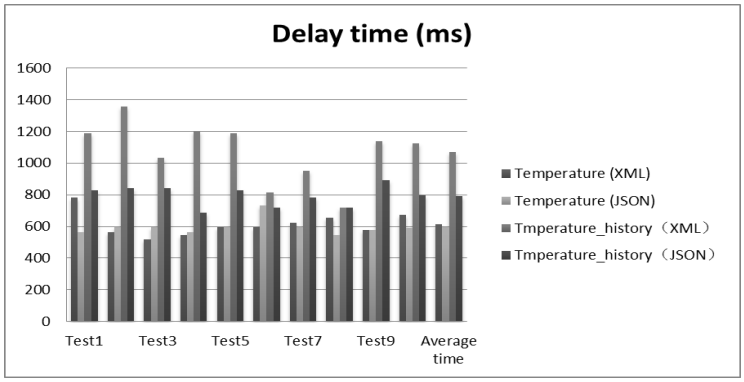

Fig. 4. Time is the duration from the creation of a request in the client to the arrival of the response back by smart gateway.

Fig. 4 presents the results of the experiment executions delay time. There are four test experiments: temperature real-time date with XML presentation, temperature real-time date with JSON presentation, temperature history date with XML presentation, temperature history date with JSON presentation. Since the temperature history back response has more than 1000 items of

data compared with the current data back response, it has longer delay time in the experiment. When the amount of data is small (real-time data response is small amount), JSON and XML presentations have more or less time delay. If the data amount is large, using JSON as resource presentation has less time delay than XML.

Further, we have divided the delay time consumed in the restful gateway into three stages, which includes the time reading from the serial-over-USB port and transformation data from Zigbee format to Ethernet format, the time stored in database, the time encapsulated data to the URL's presentation. As a compared target, the normal gateway didn't comply with REST principles, which just contains read date from serial port and data transformation. T_{total} is the total time consuming in the restful gateway, T_1 is the time that read data by the serial port and data interpret; T_2 is the time storing the data into database or cache, T_3 is the time that encapsulated data to resource presentation. We performed ten times test and gain their average time in the experiment. Fig. 5 presents the results of the experiment executions. Since the restful gateway have additional database operation, which cost most time, it had longer delay time cost than the normal gateway. After all, the performance of the gateway is acceptable.

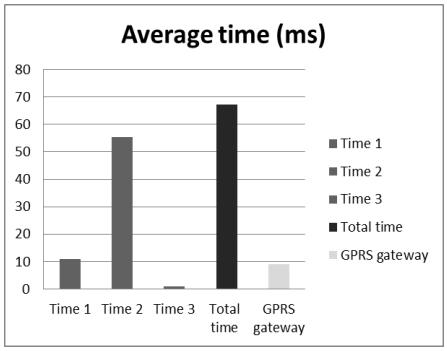

Fig. 5. Testing at each stage consuming time of the smart gateway and gateway

6. Conclusion and Future Work

In this paper we have proposed to use REST-Style architecture as a standard, to guide designing and developing of CPS. We developed a smart gateway, based on REST architectural style, which offers a uniform, efficient, and standardized way of interacting with physical devices. Flexible applications on top of heterogeneous devices can be built with little effort and acceptable performance, with any programming language that supports HTTP while Web-based solutions for traditional challenges of ad hoc environments. Through this work, our project constitutes a contribution towards the REST-style architecture for CPS.

Our future research on the REST-style architecture for CPS, besides a further evaluation about our smart gateway, is focus on some more research in device management on the system. Some advantage works in restful smart gateway for CPS are:

i. Our smart gateway abandoned the idea of having an HTTP server directly on the device. This requires a synchronization-based mechanism. Our work just simply used device management to deal with it.
ii. The REST-style architecture is most prominently the pull-based interaction model that is not suitable for modeling the communication with smart devices with respect to monitoring. Our smart gateway just periodic queries the messages to make up this drawback.

Acknowledgements. This work has been supported in part by State Key Development Program of Basic Research of China (Gant No.: 2011CB302902), Special Program for Key Basic Research of the Ministry of Science and Technology, China (Gant No.: 2011ZX03005-004-01) and National Natural Science Foundation of China (Gant No. : 61073180). We would like to thank the reviewers for their helpful comments on this paper.

References

1. Lee, EA.: Cyber physical systems: Design challenges. Isorc 2008: 11th Ieee Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing - Proceedings. 2008: 363-9
2. Prepared by the CPS Steering Group, USA. Cyber-physical Systems Executive Summary, March 6, 2008. http://varma.ece.cmu.edu/summit/CPS-Executive-Summary.pdf.
3. Rajkumar, R., Lee, I., Sha, L., Stankovic, J.: Cyber-physical systems: the next computing revolution. Proceedings of the 47th Design Automation Conference. Anaheim, California: ACM; 2010. p. 731-6.
4. Fielding, R.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine, Irvine, California, 2000.
5. Wilde, E.: Putting Things to REST. Uc Berkeley, 2007.
6. Ying, T., Vuran, M.C., Goddard, S.: Spatio-Temporal Event Model for Cyber-Physical Systems. Distributed Computing Systems Workshops, 2009 ICDCS Workshops '09 29th IEEE International Conference on; 2009 22-26 June 2009; 2009. p. 44-50.
7. Tan, Y., Goddard, S.: A prototype architecture for cyber-physical systems. SIGBED Rev. 2008; 5(1): 1-2.
8. Abdelwahed, S., Kandasamy, N., Gokhale, A.: High confidence software for cyber-physical systems. Proceedings of the 2007 workshop on Automating service quality: Held at the International Conference on Automated Software Engineering (ASE). Atlanta, Georgia: ACM; 2007. p. 1-3.
9. Chang, Y.J., Huang, W.T.: A Novel Design of Data-Driven Architecture for Remote Monitoring and Remote Control of Sensors over a Wireless Sensor Network and the Internet. J Internet Technol. 2011; 12(1): 129-37.
10. Pinto, J., Martins, R., Sousa, J.B.: Towards a REST-style architecture for networked vehicles and sensors. Pervasive Computing and Communications

Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on; 2010 March 29 2010-April 2 2010; 2010. p. 745-50.
11. Romero, D., Hermosillo, G., Taherkordi, A., Nzekwa, R., Rouvoy, R., Eliassen, F.: RESTful Integration of Heterogeneous Devices in Pervasive Environments. Distributed Applications and Interoperable Systems, Proceedings. 2010; 6115: 1-14 243.
12. Dumitrache, I.: The next generation of Cyber-Physical Systems. Control Eng Appl Inf. 2010; 12(2): 3-4.
13. Stankovic, J.A.: Cyber Physical Systems - Aspects as a Basis for Robustness and Openness. Aosd'09: 8th International Conference on Aspect-Oriented Software Development. 2009: 123-266.
14. Sztipanovits, J.: Composition of cyber-physical systems. ECBS 2007: 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, Proceedings. 2007: 3-4 614.
15. Wolf, W.: Cyber-physical Systems. Computer. 2009; 42(3): 88-9.
16. Restlet - RESTful web framework for Java, 2007. http://www.restlet.org/.
17. Crockford, D.: The application/json Media Type for JavaScript Object Notation (JSON). Internet informational RFC 4627, July 2006.
18. DeRose, S., Maler, E., and Orchard, D.: XML Linking Language (XLink) Version 1.0. World Wide Web Consortium, Recommendation REC-xlink-20010627, June 2001.
19. Luckenbach, T., Gober, P., Arbanowski, S., Kotsopoulos, A., and Kim, K.: TinyREST-a protocol for integrating sensor networks into the internet. in Proc. of REALWSN, 2005.
20. Stirbu, V.: Towards a restful plug and play experience in the web of things. Semantic Computing, 2008 IEEE International Conference on, pages 512-517, Aug. 2008.

Qiang Li received his B.S. degree in University of Nankai, Tianjin, China, in 2009. Now he is a PHD student in Institute of Software, the Chinese Academy of Sciences, Beijing, China. His interests include Cyber Physical System, wireless local network and Technologies for Next Generation of Internet.

Weijun Qin is an assistive research professor at the Institute of Software, Chinese Academy of Sciences. He completed the master and PhD in 2010 at Tsinghua University in the area of pervasive computing, and received the bachelor's degree in computer science from Tsinghua University in 2003. His research interests include wireless sensor networks, internet of things, pervasive computing and context-aware computing.

Bing Han received the M.S. degree from Beijing University, in China, in 2011. He was a visiting student at the Institute of Software, Chinese Academy of Sciences in 2010. His interests include Cyber Physical System and wireless sensor network.

Ruicong Wang received the M.S. degree from Beijing University in China, in 2011. She was He was a visiting student at the Institute of Software, Chinese Academy of Sciences in 2010. Her interests include Cyber Physical System and wireless sensor network.

Limin Sun received his B.S., M.S., and D.Sc. degree in College of Computer, National University of Defense Technology, in China, in 1988, 1995, and 1998, respectively. He is currently a Professor in Institute of Software at Chinese Academy of Sciences (ISCAS). He is also a member of IEEE and a senior member of China Computer Federation (CCF). He is an editor of Journal of Computer Science and an editor of Journal computer Applications (Chinese journals). He is also a guest editor of special issues in EURASIP, Journal of Wireless Communications and Networking, and Journal of Networks, respectively. His research interests include Mobile Vehicle Networks, Delay-Tolerant Networks, Wireless Sensor Networks, Mobile IP and Technologies for Next Generation of Internet.

Received: March 10, 2011; Accepted: May 3, 2011.